Análisis de la Robustez del Método de Asignación MATEHa*

Laura De Giusti [1], Franco Chichizola [2], Marcelo Naiouf [3], Armando De Giusti [4]
{ldgiusti,francoch,mnaiouf,degiusti}@lidi.info.unlp.edu.ar

Instituto de Investigación en Informática (III-LIDI) – Facultad de Informática – UNLP

Abstract

The TTIGHa model has been developed to model and predict the performance of parallel applications run over heterogeneous architectures.

In addition, the task assignment algorithm was implemented to MATEHa processors based on the TTIGHa model.

This paper analyzes the assignment algorithm robustness before different variations which the model parameters may undergo (basically, communication and processing times).

Keywords: *Parallel Systems. Cluster and Multi-cluster Architectures. Performance prediction models. Tasks to processors mapping. Heterogeneous Processors Robustness.*

Resumen

Se desarrolló el modelo TTIGHa utilizado para modelizar y predecir performance de aplicaciones paralelas que se ejecutan sobre arquitecturas heterogéneas.

Además, se implementó el algoritmo de asignación de tareas a procesadores MATEHa basado en el modelo TTIGHa.

En este trabajo se analiza la robustez del algoritmo de asignación frente a diferentes variaciones que pueden sufrir los parámetros del modelo (básicamente tiempos de comunicación y tiempos de procesamiento).

Palabras Clave: *Sistemas Paralelos. Arquitecturas de Cluster y Multicluster. Modelos de predicción de performance. Mapeo de tareas a procesadores. Heterogeneidad. Robustez.*

WPDP – Workshop de Procesamiento Distribuido y Paralelo

---

[1] Becaria de Formación Superior UNLP. Profesor Adjunto Facultad de Informática UNLP.
[2] Becario de Doctorado del CONICET. Profesor Adjunto Facultad de Informática UNLP.
[3] Profesor Titular D.E. Facultad de Informática UNLP.
[4] Investigador Principal CONICET. Profesor Titular D.E. Facultad de Informática UNLP.

* *Esta investigación es financiada por la CIC , la Fundación YPF y el proyecto Grid CyTED.*

Introducción

En la Ciencia Informática, los modelos de computación son usados para describir entidades reales tales como arquitecturas de procesamiento y resultan una versión "abstracta" o simplificada de la máquina física, capturando características esenciales e ignorando detalles sin importancia de la implementación [1][2][3]. Un modelo no se relaciona necesariamente con ninguna computadora real, sino que su principal razón de ser es ayudar a comprender la computación. Provee un marco para estudiar problemas, obtener ideas sobre sus distintas estructuras, y desarrollar soluciones. Una vez que un algoritmo fue diseñado para resolver un problema con un cierto modelo, éste permite dar una descripción significativa del mismo, derivar un análisis detallado e incluso predecir la performance [4] [5].

En el caso de computadoras paralelas, los requerimientos mínimos que debe cumplir un modelo es ser conceptualmente simple de entender y usar, que la determinación de corrección de un algoritmo sobre el modelo sea válida independientemente de la arquitectura física, que *la performance real se corresponda con la predicha por el modelo,* y que se aproxime a las arquitecturas reales para minimizar la brecha conceptual entre modelo y arquitectura física.
En estos requerimientos queda claro que un objetivo central de los modelos de cómputo paralelo es la posibilidad de *predicción de performance* que brinden: el éxito o fracaso dependerán en gran parte de este punto [6].

Actualmente las arquitecturas más utilizadas por su relación costo/performance son los clusters y multiclusters de procesadores, razón por la cual resulta de importancia el desarrollo de un modelo que se adecue a las características de estas plataformas. Un elemento fundamental que aparece en estas arquitecturas es la potencial *heterogeneidad* de los procesadores y las comunicaciones entre los mismos, lo cual agrega un elemento a la complejidad intrínseca de la modelización. [7][8].

En la actualidad existen diferentes modelos basados en grafos para caracterizar el comportamiento de aplicaciones paralelas en arquitecturas distribuidas [9][10][11]. Entre los modelos se pueden mencionar el modelo TIG (Grafo de Interacción de Tareas), TPG (Grafo de Precedencia de Tareas) y TTIG (Grafo de Interacción Temporal de Tareas) [12].

El problema de los modelos mencionados anteriormente es que consideran que la arquitectura en donde se ejecuta la aplicación es homogénea, situación que en general no se da en clusters y multiclusters. Por ello se ha desarrollado el modelo TTIGHa el cual considera la heterogeneidad tanto en los procesadores como en la red de comunicación [13].

Una vez definido el grafo que modela la aplicación, el problema de "mapping" se resuelve mediante algún algoritmo que establece un mecanismo automático para realizar la asignación de tareas a procesadores, y así obtener mejores resultados en la ejecución de la aplicación [14] [15] [16] [17]. Este es un problema NP-completo, debido a la existencia de gran cantidad de factores a tener en cuenta que directa o indirectamente influyen en el tiempo de ejecución del programa. Los algoritmos de mapping estático pueden clasificarse en dos grandes grupos:

- *óptimo:* se evalúan todas las posibles formas de asignar las tareas a los diferentes procesadores. Este tipo de soluciones solo puede abordarse cuando el número de configuraciones posibles es lo suficientemente bajo. En caso contrario la solución óptima no puede llevarse a cabo debido a la explosión combinatoria en el número de soluciones posibles.

- *heurístico:* se basan en técnicas de aproximación que utilizan suposiciones "realistas" del algoritmo y sistema paralelo. Dichos algoritmos producen soluciones subóptimas en tiempos de ejecución más razonables comparado con las estrategias óptimas.

Para este trabajo se utiliza el modelo TTIGHa para representar las aplicaciones a ejecutar, y el algoritmo MATEHa para resolver el problema de la asignación de los procesos a los procesadores que forman la arquitectura antes mencionada.

Modelo TTIGHa

El modelo TTIGHa se basa en la construcción de un grafo G (V,E) para representar la aplicación que se quiere modelar. Para la construcción de dicho grafo se utilizan, además de información de la aplicación, parámetros que permiten caracterizar la arquitectura (Tp,Tc), donde Tp es el conjunto de procesadores y Tc representa el conjunto de tipos diferentes de comunicación.

Los elementos que componen el grafo son:

- V, es el conjunto de nodos donde cada uno representa una tarea T_i del programa paralelo.
- E, es el conjunto de aristas que representan la comunicación entre los nodos del grafo.

Detalle de los parámetros del modelo

Tp involucra al conjunto de procesadores. Como se mencionó en la definición de la arquitectura la misma puede ser heterogénea, por lo tanto se tiene un conjunto de tipos diferentes de procesadores, y cada elemento del conjunto Tp debe indicar a qué tipo de procesador pertenece.

Tc representa al conjunto de tipos diferentes de comunicaciones. Para cada comunicación del conjunto se indica su tiempo de startup y de transferencia de un byte; esto es necesario ya que el modelo permite que la red de interconexión sea heterogénea.

En el primer parámetro del grafo (V) cada nodo representa una tarea T_i del programa paralelo. En cada nodo se almacena el tiempo de ejecución correspondiente a la tarea que representa en cada tipo de procesador.

En el segundo parámetro del grafo (E), las aristas representan cada una de las comunicaciones que existen entre cada par de tareas. En este conjunto una arista A entre dos tareas T_i y T_j mantiene una matriz C de dimensión [**m x m**] (**m:** cantidad de procesadores de la arquitectura), donde $C_{ij}(s,d)$ es el tiempo de comunicación entre la tarea T_i ubicada en el procesador s y la tarea T_j alocada en el procesador d. Es importante notar que el costo de comunicación es dependiente de los procesadores que se comunican ya que la red de interconexión se considera heterogénea. Además mantiene el "grado de concurrencia" entre la tarea T_i y la tarea T_j .

El "grado de concurrencia" es una matriz h_{ij} de dimensión [**m x m**], donde $h_{ij}(s,d)$ representa el grado de concurrencia entre las tarea T_i en el procesador s y la tarea T_j en el procesador d. Este índice (grado de concurrencia) está normalizado entre 0 y 1. Para dos tareas T_i y T_j que se comunican de T_i a T_j, el grado de concurrencia se define como el máximo porcentaje del tiempo de cómputo de T_j que puede ser realizado en paralelo con T_i, teniendo en cuenta sus dependencias mutuas provocadas por las comunicaciones existentes entre ambas tareas, y sin contemplar el costo de comunicación asociado a las mismas (esto genera un valor independiente a los datos a transmitir).

Algoritmo MATEHa

Este algoritmo permite determinar la asignación de tareas a los procesadores de la arquitectura a utilizar buscando minimizar los tiempos de ejecución de la aplicación en dicha arquitectura. MATEHa considera una arquitectura con un número acotado de procesadores, que pueden ser heterogéneos en cuanto a su potencia de cálculo y a la red de interconexión.

La estrategia de MATEHa consiste en determinar, para cada una de las tareas del grafo G formado por el modelo TTIGHa, a qué procesador debe ser asignada para lograr el mayor rendimiento de la aplicación en la arquitectura utilizada. Dicha asignación usa los valores generados en la construcción del grafo: tiempo de cómputo de una tarea en cada procesador, tiempo de comunicación con sus adyacentes (el cual también depende de donde han sido asignadas las tareas) y por último el grado de paralelismo entre tareas. Este último valor es útil para tomar la decisión de asignar al mismo procesador aquellas tareas con menor grado de paralelismo, o asignar a procesadores diferentes aquellas tareas que pueden ejecutarse en forma concurrente (el valor de su grado de paralelismo es alto).

El algoritmo de mapping extrae los valores mencionados anteriormente del modelo TTIGHa, en los que se fundamenta la heurística de asignación del algoritmo. En primer lugar, para cada nodo del grafo del modelo TTIGHa se define el *nivel* que será el que se utilice para realizar la asignación de las tareas del grafo con cierta prioridad.

En segundo lugar, para cada nivel n del grafo (comenzando del nivel 0), se realiza la asignación de todas sus tareas a los procesado. Para esto, en cada paso se elige aquella tarea aún no asignada perteneciente al nivel n que genera la máxima ganancia al asignar dicha tarea a un procesador. La ganancia de una tarea Ti se obtiene como la diferencia entre el costo de ejecutar Ti en el “peor procesador” y la ejecución de Ti en el “mejor procesador” (esto no implica que el mejor / peor procesador sea el más rápido / lento respectivamente).

Para calcular el costo c de ejecutar una tarea Ti en un procesador p se realizan dos acciones. La primera suma al tiempo acumulado en p (este tiempo es la suma de los tiempos de ejecución de las tareas ya asignadas a él) el tiempo requerido para ejecutar Ti en p. En la segunda, por cada tarea Ta adyacente a Ti, que ya ha sido asignada a un procesador q (diferente a p) se acumula al costo c tiempo de comunicación entre Ti y Ta (en ambos sentidos) y el tiempo en que Ti y Ta no pueden ejecutarse de manera conjunta, es decir, el porcentaje en que no se ejecutan concurrentemente (1-grado de concurrencia entre Ti y Ta) multiplicado por el tiempo de ejecutar Ta en q.

Contribución de este trabajo

En el diseño de algoritmos automáticos de mapping, una característica importante es su robustez, ya que los algoritmos robustos permiten encontrar una asignación adecuada a pesar de basarse en datos aproximados (tiempos de cómputo y comunicación de las tareas) a los valores reales de la aplicación [18] [19].

En este trabajo se estudia dicha característica en el algoritmo de asignación MATEHa desarrollado para el modelo TTIGHa. El estudio se realiza mediante un conjunto de pruebas que analizan el comportamiento del algoritmo frente a variaciones en los parámetros de entrada del mismo (tiempos de ejecución y de comunicación entre las tareas).

Trabajo Desarrollado

Como se mencionó en la sección anterior se realizaron diferentes pruebas experimentales para probar la robustez del algoritmo. Para la experimentación se realizaron los siguientes pasos:

1- Elegir la arquitectura para las pruebas.
2- Elegir el conjunto de aplicaciones con distintas características a evaluar.
3- Generar la asignación para cada una de las tareas de cada aplicación utilizando el algoritmo MATEHa.
4- Probar la robustez de cada una de estas asignaciones.

Elegir la arquitectura para las pruebas

La arquitectura heterogénea utilizada está compuesta por dos clusters conectados mediante un switch. El primero de ellos está formado por 20 procesadores cada uno de 2.4 Ghz Pentium IV con 1G RAM (denominado de acá en adelante como *cluster 1*). El segundo está compuesto por 10 procesadores, donde cada uno de ellos es de 2 GHz Celeron con 128 M RAM (llamado *cluster 2* de acá en más). La conexión entre los procesadores dentro de cada cluster es a través de una red Ethernet de 100 Mbits.

Esta arquitectura fue elegida de manera que los clusters que la componen sean de características diferentes en cuanto a la potencia de cálculo de los procesadores.

Para las pruebas se escogieron diferentes subconjuntos de procesadores de cada uno de los clusters, formando cuatro configuraciones:

- Configuración 1: 4 procesadores pertenecientes al cluster1.
- Configuración 2: 3 procesadores pertenecientes al cluster1 y 1 perteneciente al cluster2.
- Configuración 3: 2 procesadores pertenecientes al cluster1 y 2 pertenecientes al cluster2.
- Configuración 4: 1 procesador perteneciente al cluster1 y 3 pertenecientes al cluster2.

Elegir el conjunto de aplicaciones diferentes a evaluar

Se eligió un conjunto diferente de aplicaciones. Cada una de éstas variaba en cuanto a cantidad de tareas de la aplicación, tamaño de las tareas, cantidad de subtareas que componen una tarea y volumen de las comunicaciones entre subtareas. Todas estas características deben ser configuradas para cada aplicación.

En cada una de las pruebas realizadas para las diferentes aplicaciones se debe indicar primero la configuración de la arquitectura a utilizar. Una vez que se ha elegido la arquitectura debe especificarse los tipos diferentes de procesadores, la cantidad de procesadores para cada uno de estos tipos, los tipos diferentes de comunicación, el tiempo de startup y transferencia para cada uno de estos tipos, y por último el tipo de comunicación utilizado entre cada par de procesadores.

Una vez que esta información fue especificada se crea el grafo G generado a partir del modelo TTIGHa.

Generar la asignación para cada una de las tareas de cada aplicación utilizando el algoritmo MATEHa.

Utilizando el grafo generado para cada prueba en el punto anterior, se realiza la asignación mediante el algoritmo MATEHa y se lo compara con la asignación óptima (que se obtiene

analizando cómo funciona el algoritmo para todas las posibles asignaciones de tareas a procesadores). Esta comparación se realiza para poder determinar la eficacia de la asignación del algoritmo en cada una de las diferentes pruebas.

A modo de comentario en este punto se puede citar los resultados de un trabajo previo en el cual para las pruebas descriptas anteriormente el porcentaje de diferencia frente a la asignación óptima no supera el 12%.

Probar la robustez de cada una de estas asignaciones

La robustez de un algoritmo está relacionada con la sensibilidad de las variaciones en la estimación de los parámetros de entrada del modelo. Para el modelo utilizado, los parámetros que pueden no ser exactos al momento de calcular la asignación son: tiempo de ejecución de cada tarea en cada tipo diferente de procesador y los diferentes tiempos de comunicación dependientes de la red utilizada para la misma.

Dado que se utiliza un mapping estático, se presupone que las variaciones en los parámetros mencionados anteriormente son pequeñas y por lo tanto se estudia la influencia de las mismas en el algoritmo de asignación MATEHa.

Para concluir sobre el grado de robustez del algoritmo se analiza la sensibilidad del mismo con respecto al tiempo de ejecución de las tareas y a los diferentes tiempos en las comunicaciones.

Para analizar la sensibilidad del algoritmo MATEHa frente a posibles variaciones en sus parámetros de entrada se realizan una serie de pruebas incluyendo diferentes porcentajes de “ruido” (variaciones). Estas pruebas consideraron tres tipos de variaciones: sólo en los tiempos de cómputo de las tareas, sólo en los tiempos de las comunicaciones, y variaciones en los tiempos de cómputo de las tareas y en las comunicaciones de las mismas.

Experimentación Realizada

Para cada una de las diferentes aplicaciones definidas en el punto 2, las cuales consideraban distintas características en cuanto a las tareas que componían la aplicación y las comunicaciones, se realizaron pruebas agregando diferentes porcentajes de variaciones en el tiempo de cómputo y/o comunicación. Cada variación que se considera es un valor aleatorio entre 0 y un porcentaje máximo (diferente según cómputo o comunicación). Los valores para el porcentaje máximo tenidos en cuenta son del 0 al 100% en intervalos de 10%. Para obtener una muestra más significativa se generaron 10 corridas para cada una de estas variaciones.

En cada prueba se realizaron los siguientes pasos:

a) Para la aplicación a ejecutar se calcula el modelo TTIGHa.
b) Se obtiene la asignación (por medio del algoritmo de mapping MATEHa) para esa aplicación de acuerdo a los tiempos indicados en la prueba.
c) Se calculan los nuevos tiempos de cómputo y/o comunicación sumándole el porcentaje de variación correspondiente.
d) Con la asignación obtenida en b) y los nuevos tiempos calculados en c) se genera la simulación de la ejecución de la aplicación, para obtener el tiempo final.
e) Con los tiempos obtenidos en b) se calcula la asignación, también utilizando MATEHa y luego se realiza la simulación para dicha asignación.

f) Se comparan los tiempos finales obtenidos por las simulaciones de los puntos d) y e). Cuanto más cercanos son dichos tiempos implica que la asignación lograda por el algoritmo MATEHa se ve poco afectado por variaciones en los tiempos del modelo.

Resultados Obtenidos

Para poder analizar los resultados obtenidos, a partir de los mismos se calcula para cada uno de los distintos porcentajes (0..100%) de variación:

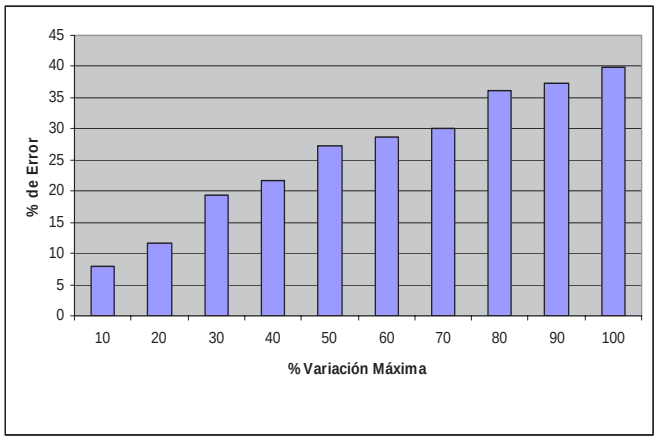
- Porcentaje de pruebas en las que hubo error, es decir, en las que el tiempo final obtenido en los puntos d) y e) fue diferente (% con Error).
- Error promedio. El error en una prueba está dado por la diferencia entre los tiempos obtenidos en d) y e) respecto al tiempo obtenido en 5 (Error Promedio General).
- Error promedio de las pruebas que tuvieron resultados diferentes en d) y e), este valor se calculó para realizar un análisis mas detallado de cuanto era la influencia del error en los resultados (Error Promedio Podado).

La Tabla 1 muestra algunos de los resultados obtenidos para las pruebas en donde las variaciones son en los tiempos de cómputo y comunicación. El conjunto completo de resultados se encuentra en [20]

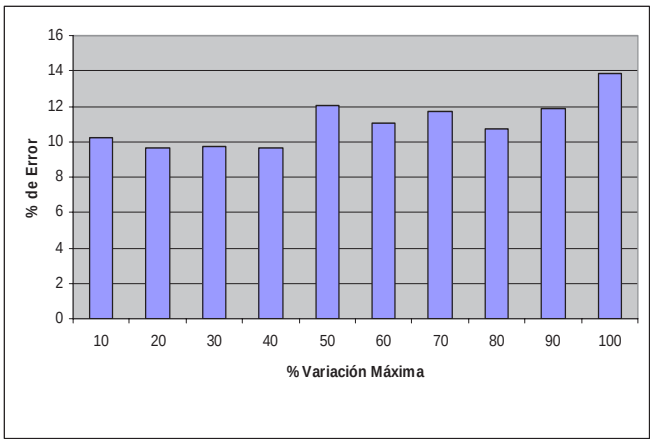
% de Variaciones (Cómputo-Comunicación)	% con Error	Error Prom. General	Error Prom. Podado
0-10	10,237	0,003	0,037
0-20	9,687	0,002	0,028
0-30	9,765	0,002	0,029
0-40	9,609	0,003	0,032
0-50	12,031	0,003	0,029
0-60	11,015	0,003	0,033
0-70	11,718	0,004	0,034
0-80	10,703	0,003	0,031
0-90	11,875	0,004	0,039
0-100	13,828	0,004	0,032
10-0	7,968	0,003	0,042
10-10	16,95	0,005	0,032
10-20	17,97	0,006	0,033
10-30	19,77	0,006	0,032
10-40	15,78	0,005	0,036
10-50	16,02	0,006	0,042
10-60	18,28	0,007	0,038
10-70	18,05	0,005	0,031
10-80	18,44	0,005	0,028
10-90	17,89	0,005	0,031
10-100	18,59	0,006	0,036
40-0	21,640	0,009	0,042
40-10	27,34	0,011	0,043
40-20	29,14	0,010	0,034
40-30	31,33	0,011	0,035
40-40	30,08	0,011	0,037
40-50	30,00	0,012	0,040
40-60	29,92	0,012	0,041
40-70	30,39	0,010	0,033
40-80	32,42	0,012	0,038
40-90	29,30	0,010	0,034
40-100	29,45	0,010	0,034
60-0	28,75	0,013	0,047
60-10	34,77	0,016	0,048
60-20	35,47	0,015	0,043
60-30	32,42	0,014	0,045
60-40	33,83	0,015	0,044
60-50	36,88	0,015	0,042
60-60	35,94	0,015	0,043
60-70	34,61	0,015	0,043
60-80	35,16	0,015	0,044
60-90	33,91	0,013	0,039
60-100	33,75	0,012	0,037
100-0	39,765	0,022	0,056
100-10	40,23	0,022	0,056
100-20	42,50	0,021	0,051
100-30	38,44	0,020	0,052
100-40	42,42	0,021	0,049
100-50	43,91	0,023	0,052
100-60	43,20	0,024	0,056
100-70	41,33	0,022	0,055
100-80	41,25	0,023	0,056
100-90	43,83	0,018	0,043
100-100	45,39	0,023	0,052

Tabla 1. Resultados de las pruebas con variaciones en los tiempos de cómputo y comunicación

La Figura 1a) muestra el porcentaje de error obtenido para los diferentes valores en las variaciones de cómputo, mientras que la Figura 1b) presenta el porcentaje de error obtenido para los diferentes valores en las variaciones de las comunicaciones.



a) Porcentaje de error con diferentes variaciones en el cómputo

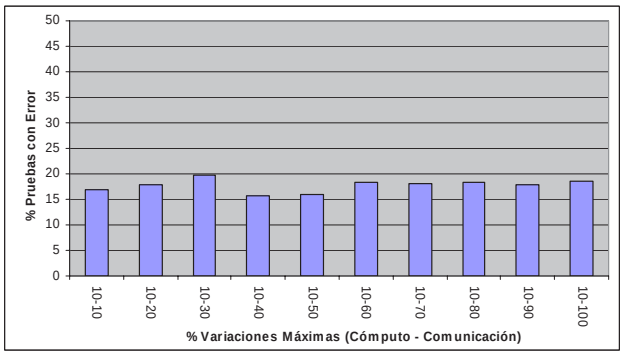


b) Porcentaje de error con diferentes variaciones en la comunicación

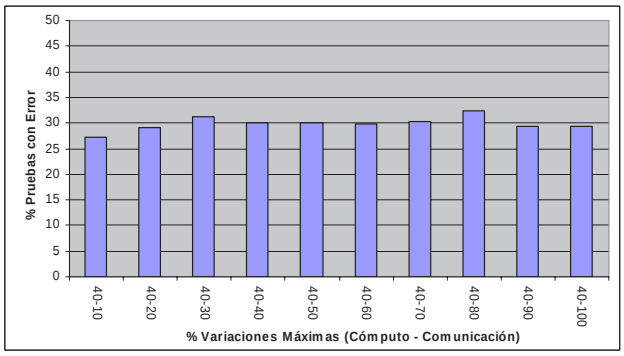
Figura 1

De las Figuras 1a) y 1b) se puede notar que al variar el porcentaje de cómputo esto genera que el error aumente, sin embargo no ocurre en la misma medida con el porcentaje de error cuando se varían los valores en las comunicaciones.

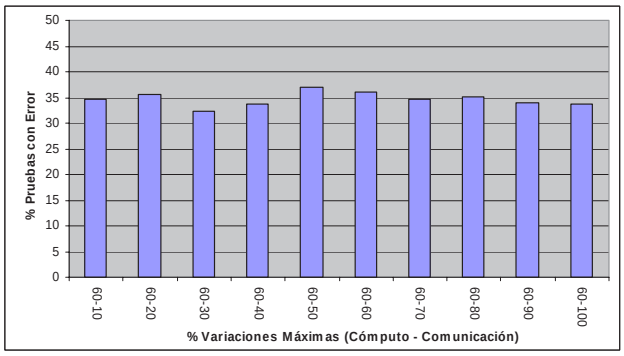
La Figura 2 muestra un conjunto de gráficos que consideran variaciones tanto en el cómputo como así también en las comunicaciones.



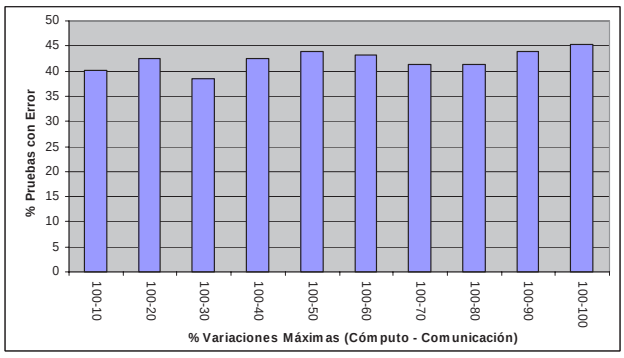
Porcentaje de pruebas con error con
10 % de variación en el cómputo.



Porcentaje de pruebas con error con
40 % de variación en el cómputo.



Porcentaje de pruebas con error con
60 % de variación en el cómputo.

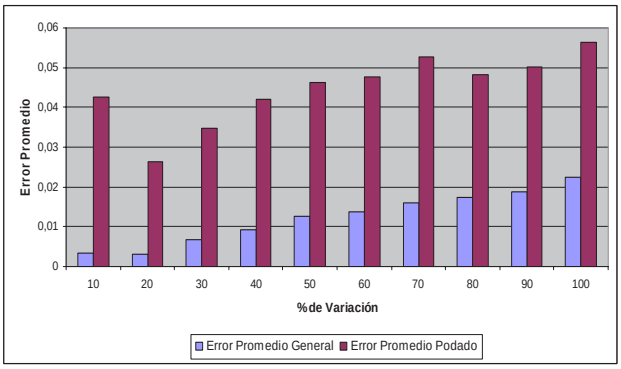


Porcentaje de pruebas con error con
100 % de variación en el cómputo.

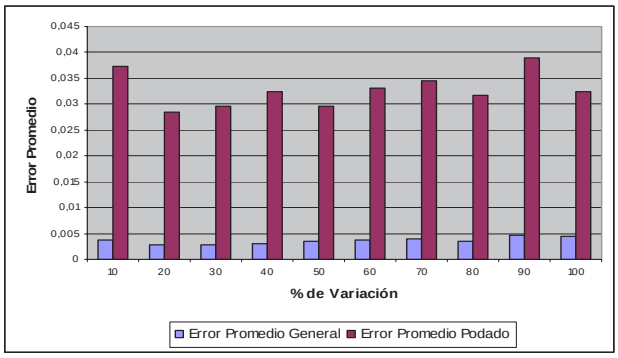
Figura 2

Al combinar las variaciones tanto en los tiempos de cómputo como en los de comunicaciones, se puede ver que respecto a los porcentajes de pruebas donde se detectó errores mantiene los lineamientos de cuando solo se analizó las variaciones en el cómputo, pero con un leve aumento.

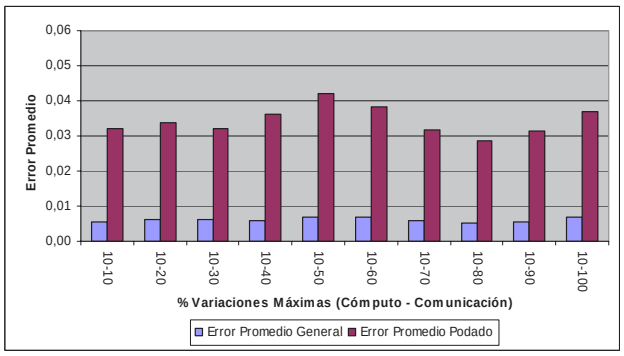
Como se describió anteriormente también se analizó el promedio de error cometido en las pruebas en las que hubo error. La Figura 3 permite observar estos promedios.



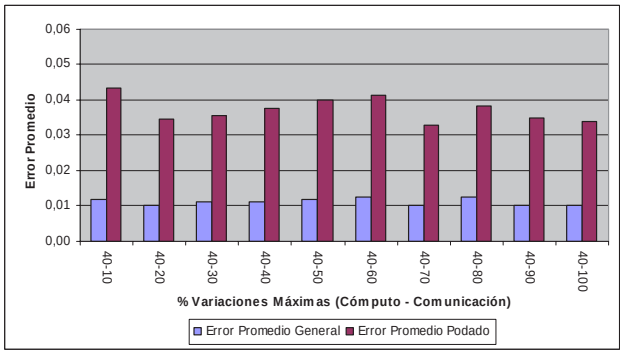
Porcentaje de error general y podado utilizando diferentes variaciones de cómputo



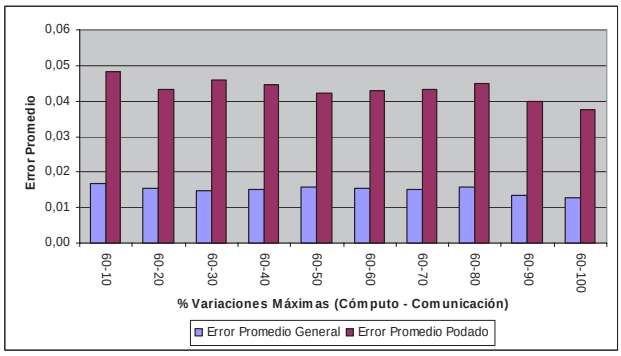
Porcentaje de error general y podado utilizando diferentes variaciones de comunicación



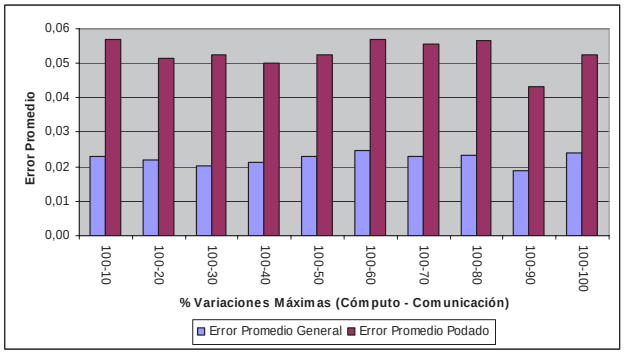
Error promedio general y podado con 10 % de variación en el cómputo.



Error promedio general y podado con 40 % de variación en el cómputo.



Error promedio general y podado con 60 % de variación en el cómputo.



Error promedio general y podado con 100 % de variación en el cómputo.

Figura 3

La Figura 3 permite observar el error promedio general y el podado para las combinaciones de porcentajes de variación evaluados. En cada gráfico se mantiene fijo el porcentaje de variación respecto al cómputo, variando el de comunicación. En las mismas se puede ver que para todas las

variaciones el porcentaje de error podado no supera el 6%, mientras que el porcentaje de error general no supera el 2.5%.

Al igual que con el porcentaje de pruebas con error, al combinar las variaciones tanto en los tiempos de cómputo como en los de comunicaciones, se observa que tanto el error promedio general como el podado mantienen la forma de cuando se analizaron sólo variaciones en el cómputo, es decir aumentan a medida que se incrementa el porcentaje de variación del tiempo de computo.

Conclusiones

La variación en el cómputo tiene influencia sobre la asignación a realizar por el algoritmo, ya que al aumentar dicha variación también aumenta el porcentaje de error. Sin embargo en las pruebas realizadas anteriormente se puede notar que utilizando una variación de hasta el 60% el porcentaje de pruebas con error no supera el 30%.

En las pruebas que se realizaron sólo con variaciones en los tiempos de las comunicaciones se puede observar que el porcentaje de error respecto del mapeo óptimo no supera el 14% aún haciendo variaciones del 100%. Además los gráficos también permiten ver que en general el error se mantiene prácticamente constante.

En las pruebas en las cuales se realizaron variaciones tanto en los tiempos de cómputo como en las comunicaciones se puede observar que, en cuanto al porcentaje de pruebas con error, se mantiene lo que ocurre al variar uno de los dos tipos de tiempo; resulta creciente de acuerdo a la variación en los tiempos de cómputo y se mantiene constante en relación a las variaciones de los tiempos de comunicación. Es decir que conserva la forma de cuando sólo se varía el tiempo de cómputo pero con un leve incremento relativamente constante por variar el tiempo de comunicación. En este caso, se llega a un 37 % de error al usar una variación de 60% en el cómputo.

Con respecto al error promedio podado, se observa un leve crecimiento a medida que aumenta la variación en el tiempo de cómputo, sin embargo en ningún caso supera el 6 %. Por ultimo, en el error promedio general ocurre lo mismo, sin superar el 2.5 %. Estos resultados permiten concluir que el algoritmo MATEHa presenta un alto grado de robustez, ya que logra realizar una buena asignación sin la necesidad de utilizar los parámetros exactos en cuanto a tiempo de cómputo y comunicación.

Bibliografía

[1] Grama A., Gupta A., Karypis G., Kumar V., "An Introduction to Parallel Computing. Design and Analysis of Algorithms", Pearson Addison Wesley, 2nd Edition, 2003
[2] Hagit Attiya, Jennifer Welch, Distributed Computing: Fundamentals, Simulations, and Advanced Topics (Wiley Series on Parallel and Distributed Computing). Wiley-Interscience; 2 edition (March 12, 2004).
[3] Kenneth A. Berman, Jerome L. Pau, Algorithms: Sequential, Parallel, and Distributed. Course Technology; 1 edition (October 11, 2004).
[4] Leopold C., "Parallel and Distributed Computing. A survey of Models, Paradigms, and Approaches", Wiley Series on Parallel and Distributed Computing. Albert Zomaya Series Editor, 2001
[5] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. Thee Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Massachusetts, 1974
[6] Akl S, “Parallel Computation. Models and Methods”, Prentice-Hall, Inc., 1997.

[7] Bohn C., Lamont G., "Load Balancing for Heterogeneous Clusters of PCs", Future Generation Computer Systems, Elsevier Science B.V., Vol 18, 2002, pp 389-400
[8] Goldman. “Scalable Algorithms for Complete Exchange on Multi-Cluster Networks”. CCGRID'02, IEEE/ACM, Berlin, p. 286 - 287, 2002.
[9] C. Roig, A. Ripoll, M.A. Senar, F. Guirado, and E. Luque. Modelling Message-Passing Programas for Static Mapping. In Euromicro Workshop on Parallel and Distributed Processing (PDP’00). IEEE CS Press. USA, pp 229-236, 1999
[10] J.J. Hwang, Y.C. Chow, F.D. Anger, and C.Y. Lee. Scheduling Precedence Graphs in Systems with Interprocessor Communication Times. SIAM Journal of Computing, 18(2): 244-257, April 1989
[11] A. Kalinov, S. Klimov. Optimal Mapping of a Parallel Application Processes onto Heterogeneous Platform. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS’05), April 2005
[12] C. Roig, “Algoritmos de asignación basados en un nuevo modelo de representación de programas paralelos”, Tesis Doctoral, Universidad Autónoma de Barcelona, 2002.
[13] Laura De Giusti, Franco Chichizola, Marcelo Naiouf, Ana Ripio, Armando De Giusti, “ A Model for the Automatic Mapping of Task to Proccessors in Heterogeneous Multicluster Architecture”, Journal of Computer Science and Technology , Vol 7 nro 1 ISSN: 1666-6046, pag 39-44, Marzo 2007 .
[14] J. Cuenca, D. Gimenez, and J. Martinez, “Heuristics for Work Distribution of a Homogeneous Parallel Dynamic Programming Scheme on Heterogeneous Systems”, Procss of the 3rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (HeteroPar’04), July 5-8, 2004 Cork, Ireland, IEEE CS Press.
[15] M. Garey and D. Johnson. Computers and Intractability. W.H. Freeman and Co. S. Francisco, 1979
[16] J.C. Cunha, P. Kacksuk, and S.C. Winter. Parallel Program development for cluster computing. Nova Science Pub. Inc., Huntington, New York, 2001
[17] C. Roig, A. Ripoll, M. Senar, F. Guirado, and E. Luque. “Exploiting knowledge of temporal behavior in parallel programs for improving distributed mapping”. 6th. International Euro-Par conference. Lecture Notes in Computer Science,1900:262–271, 2000.
[18] Ali, S., Maciejewski, A.A., Siegel, H.J., Kim, J.-K., 2003. Definition of a robustness metric for resource allocation. In: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS). p. 42.
[19]Darin England, Jon Weissman, and Jayashree Sadagopan, A New Metric for Robustness with Application to Job Scheduling,
IEEE International Symposium on High Performance Distributed Computing 2005 (HPDC-14), July 24-27, 2005, Research Triangle Park, NC.
[20] Franco Chichizola, Laura De Giusti. Reporte Técnico: “Pruebas Realizadas para al algoritmo de mapping MATEHa asignando Variaciones de Cómputo y Comunicación a los parámetros del modelo TTIGHa”, Marzo 2007.